baterm

An interactive shell for BATEM building-energy simulation

A short, live demo

What is `baterm`?

- An interactive `cmd2` shell on top of the `batem` Python package
- One coherent workflow around four artifacts:
  - **`ContextData`** — site, weather, year, side masks
  - **`BuildingData`** — footprint, envelope, glazing, HVAC, shading
  - **`Building`** — simulation-ready combination of context + building
  - **`PVplantData`** — optional PV array (shares the session `DataProvider`)
- Reproducible batches via `.baterm` script files
- All heavy / GUI work is delegated to **child Python processes**
  → the shell stays responsive

Mental model

```
flowchart LR
  ContextData["ContextData"] --> Building
  BuildingData["BuildingData"] --> Building
  ContextData --> PV["PVplantData"]
  Building --> bsim["building simulate"]
  PV --> psim["pv simulate"]
  bsim --> enercost
  psim --> enercost
  bsim --> variation
  psim --> balance["pv balance / results"]
```

Each verb in `baterm` operates on one of those boxes. Shared series live on one `DataProvider`.

Commands at a glance

Family	Top-level command	help …
Site / climate	`context`	`help context`
Thermal model	`building`	`help building`
PV array	`pv`	`help pv`
Distant shading	`sidemask`	`help sidemask`
Materials DB	`material`	`help material`
Electricity bill	`enercost`	`help enercost`
Workspace	`cd` `pwd` `ls` `dir` `status` `clear` …	`help baterm`
Scripts	`run` `pyrun`	`help run` / `help pyrun`
Docs / notebooks / slides	`manual` `workshops` `lectures` `naming`	`help manual` / `help workshops` / `help lectures`

Type `help` or `help baterm` in the shell for the same map.

Commands — `context`

Command	Role
`context`	Summary of loaded `ContextData`
`context new` / `edit`	GUI create / modify
`context load` / `save` / `delete`	File & memory
`context 3d` / `heliodon`	Side masks & sun paths
`context best angles`	Best exposure / slope (irradiation)
`context temperature` `rain` `solar` `wind`	Climate figures
`context psychro` `heatwave` `evolution`	Comfort / extremes / trends
`context plot` / `vars`	Context-origin series

Commands — building

Command	Role
`building`	Compact `BuildingData` summary
`building new` / `edit`	GUI create / modify
`building load` / `save` / `delete`	File & memory
`building footprint` `3d` `heliodons`	Geometry views
`building simulate [--quick]`	Hourly HVAC + comfort
`building results` `plot` `vars`	Results & series
`building variation …`	Parametric sweeps

Materials: `material list thermal` · `material list glazing` · `material thermal load <name> <row>`

Commands — `pv`

Command	Role
`pv`	Summary of loaded `PVplantData`
`pv new` / `edit`	GUI (needs a session context)
`pv load` / `save` / `delete`	File & memory
`pv dir [folder]`	List `.pv` files
`pv 3d`	Array layout (3D)
`pv best angles`	Best exposure / slope for DC yield
`pv simulate [--quick]`	Hourly `PV:power_W` on `DataProvider`
`pv results`	Production + building–PV indicators
`pv balance`	Day×month heatmap PV – PELEC
`pv heliodon` `plot` `vars`	Site sun path & PV series

Commands — `sidemask` · `enercost`

Command	Role
`sidemask`	Table of session + context masks
`sidemask new` / `edit <name>`	GUI create / modify
`sidemask load` / `save` / `delete`	`.sidemask` files
`sidemask 3d [name …]`	3D preview
`enercost`	Annual bill (default HP/HC)
`enercost base` / `hp_hc`	Tariff choice
`enercost … --base/--hp/--hc/--export`	Price overrides (€/kWh)

`enercost` needs **both** `building simulate` and `pv simulate`.

Commands — workspace & scripts

Command	Role
`cd` `pwd` `ls`	Navigate / list files
`dir [folder]`	List `.context` `.building` `.pv` `.baterm`
`status`	Session snapshot
`clear`	Forget context / building / PV in memory
`export <name>`	Write all series → `<name>.xlsx`
`vars` / `plot`	All origins (vs scoped `building` / `context` / `pv`)
`load <path>`	Remember `current_file` (informational)
`run <path>`	`.baterm` batch or `.sh` script
`pyrun <path.py>`	External Python (same interpreter)
`workshops` / `list` / `stop` / `<id>`	JupyterLab for `workshopXX_*.ipynb` (project root; prints token)
`lectures` / `list` / `<id>`	List / open `lectures/slidesXX_*.pdf`
`manual`	Open `doc/Baterm_User_Manual.md`

Starting the shell

```
python -m batem.baterm
or just
baterm (after `pip install batem`)
```

```
Welcome to BATEM Terminal (baterm). Type help or ? for commands.
Topics: help baterm | help context | help building | help pv | help naming
baterm > help building
```

- Tab completion on commands, subcommands and paths
- `set timing False` to hide the per-command `Elapsed: …` line
- `exit` / `quit` to leave

Workspace files

`baterm` recognises five kinds of project files:

Extension	Holds	Created by
`.context`	`ContextData` (location, year, …)	`context new/edit/save`
`.building`	`BuildingData` (geometry, systems)	`building new/edit/save`
`.pv`	`PVplantData` (+ nested context)	`pv new/edit/save`
`.sidemask`	`SideMaskData` (horizon obstacles)	`sidemask new/edit/save`
`.baterm`	Batch of baterm commands	a plain text editor

List them with `dir` (uses cwd by default):

```
baterm > dir
```

Loading a context

```
baterm > context load grenoble_bis
Context loaded: /…/grenoble_bis.context

baterm > status
Current file: None
Context location: Grenoble
Weather: loaded (context-only DataProvider).
Side masks (session): 0
```

- `.context` extension is appended automatically
- Weather is read once, cached on the active `DataProvider`

Climate visualisation

Once a context is loaded, explore the site climate:

```
baterm > context temperature   # outdoor T° time series + monotone
baterm > context rain          # precipitation & cloudiness
baterm > context solar         # unit-area solar gains by orientation
baterm > context wind          # wind rose + distributions
baterm > context psychro       # Givoni outdoor psychrometric diagram
baterm > context heatwave      # 3-day felt-temperature extremes
baterm > context evolution     # multi-year archive trends
baterm > context plot          # generic time-series picker
```

Figures pop up in a child process; the shell stays available.

Site geometry: heliodon & 3D

```
baterm > context heliodon      # sun paths + horizon + side masks
baterm > context 3d            # 3D view of all session side masks
```

Side masks are managed independently:

```
baterm > sidemask new          # GUI editor
baterm > sidemask save mountain
baterm > sidemask load mountain
baterm > sidemask 3d
```

They are reused automatically by `context heliodon` and the building views.

Loading a building

```
baterm > building load H
Building loaded: /…/H.building

baterm > status
Context location: Grenoble
Model: full Building (simulation-ready).
```

Once both **context** and **building** are loaded, `baterm` builds a real `Building` instance and exposes the full simulation surface.

Inspect the building

```
baterm > building              # summary of BuildingData fields
baterm > building footprint    # plan-view footprint
baterm > building 3d           # interactive 3D building view
baterm > building heliodons    # one heliodon per glazed facade
```

> `building 3d` opens in a separate Python process, so quitting the 3D window does **not** kill your `baterm` session.

Modeling a contact in batem

A contact is a wall whose one face has a known temperature

One JSON object per contact, inside `BuildingData.wall_contacts`:

```
[ { "vertex_id": 2, "temperature_C": 12.0, "n_floors": 3 } ]
```

Key	Meaning
`vertex_id`	Footprint edge `i` → `i+1`.
`temperature_C`	Imposed temperature (°C) on the outer face.
`n_floors`	Number of storeys spanned upward from `base_elevation`.

`null` or `[]` ⇒ no contact.

How to declare a contact in `baterm`

```
baterm > context load grenoble_bis
baterm > building load H
baterm > building footprint        # confirm the right vertex_id
baterm > building edit             # paste JSON in 'wall_contacts'
baterm > building save H_with_contact
baterm > building 3d               # black patch on the contact face
```

> Same `inout_wall` composition · no glazing · no solar on that strip.

Passive solar protection on glazing

Per-façade overhang / brise-soleil in `BuildingData.max_unshaded_protections` (JSON in `building new` / `building edit`):

```
[
  {"altitude_max_deg": 45.0, "azimuth_half_width_deg": 90.0},
  {"altitude_max_deg": 90.0, "azimuth_half_width_deg": 90.0}
]
```

- **90°** altitude or azimuth ⇒ no shading on that axis (default template in `building new`)
- Side index = footprint edge `i` → `i+1` (same as `building footprint`)

Sweep in baterm:

```
baterm > building variation solar_protection side 0 --quick
```

Simulate

```
baterm > building simulate
[…]
Total heating needs: 9821 kWh
Total cooling needs:  734 kWh
Comfort indicator (last floor): …

baterm > building results     # reprint results
baterm > building plot        # pick any DataProvider series to plot
```

- `building simulate` runs an annual hourly simulation
- All resulting series are attached to the same `DataProvider`
- `building plot` re-uses it to draw any combination of variables

Parametric sweeps — `building variation`

```
baterm > building variation --quick
baterm > building variation solar_protection side 0
baterm > building variation glazing_ratio side 0 floor 2
```

- Runs annual simulations while sweeping design parameters
- Dual-axis figures (PNG under `.baterm/variation_figures/`):
  - **Left — building** HVAC totals (heating, cooling, total) [kWh/year]
  - **Right** — comfort on reference floor: **10%** lowest / highest occupied temperatures [°C]
- Spline interpolation + exact sampled points

Useful flags:

Flag	Effect
`--quick`	Cap at 8 cases for fast iteration
`--max-jobs N`	Hard cap on total cases
`--max-workers M`	Parallel processes used

What `building variation` produces

- **One figure per scalar parameter** (`heat_recovery` , `solar_factor` , setpoints, …)
- **One figure per glazing sweep** (`side_glazing_ratios`)
- **One figure per solar-protection sweep** (`altitude_max_deg` on one façade)
- All PNGs are saved next to the active workspace:

```
.baterm/variation_figures/*.png
```

On macOS the output folder opens automatically in Finder.

Photovoltaics — load & inspect

```
baterm > context load grenoble_bis
baterm > pv load my_array          # or: pv new  (GUI; needs context)
baterm > pv                        # compact plant summary
baterm > pv 3d                     # array layout
baterm > pv best angles            # max annual DC yield angles
baterm > pv heliodon
```

- A `.pv` file embeds its own `ContextData` (loading it also sets the session context)
- `pv new` uses the **session** context; array geometry is edited in the PV form
- Same `DataProvider` as the building after simulations

Photovoltaics — simulate & KPIs

```
baterm > building simulate          # PELEC:building#sim
baterm > pv simulate                # PV:power_W  (+ collectors)
baterm > pv results                 # production + self-consumption / …
baterm > pv balance                 # day×month heatmap PV − PELEC
baterm > pv plot                    # PV-origin series
baterm > enercost                   # annual bill (HP/HC by default)
baterm > enercost base --base 0.25 --export 0.12
baterm > export annual_run          # Excel hand-off
```

Need	Command
Fast check	`pv simulate --quick` (first 7 days)
Building–PV KPIs	`pv results` (both sims required)
Surplus / deficit map	`pv balance`
€ bill	`enercost`

Replay a session — `.baterm` scripts

`test.baterm` — three lines, one command per line:

```
context load grenoble_bis
building load H
building simulate
```

Replay it from the shell:

```
baterm > run test
Running baterm script: /…/test.baterm
…
```

- `.baterm` is appended automatically if you omit the extension
- Lines starting with `#` are treated as comments
- Ctrl+C cleanly stops the batch

Three flavours of "run"

Command	What it runs	Cwd
`run <file>`	A `.baterm` batch (or `.sh` shell)	repo root
`pyrun <file.py>`	A standalone Python file	package directory
`run echo <text>`	Built-in mini commands	n/a

Aliases (set in `Baterm.__init__`):

- `script` / `run_script` / `!` → `run`
- `pyscript` / `run_pyscript` → `pyrun`

Productivity tips

- **Tab completion** — try `building <TAB>`, `pv si<TAB>`, `context te<TAB>`
- **`set timing False/True`** — toggle per-command elapsed time
- **`status`** — snapshot of context / building / PV / masks
- **`dir`** — list `.context` / `.building` / `.pv` / `.baterm`
- **`clear`** — reset memory without deleting files
- **`export`** then **`pyrun`** — hand results to external Python
- **`history`** — re-run with `!42`, `!-1`, …
- **`help <cmd>`** / **`manual`** — shell help + full Markdown manual

A 90-second demo flow

```
baterm > dir
baterm > context load grenoble_bis
baterm > context temperature
baterm > building load H
baterm > building 3d
baterm > building simulate
baterm > pv load my_array
baterm > pv simulate
baterm > pv results
baterm > enercost
baterm > building variation solar_protection side 0 --quick
baterm > run test
baterm > status
baterm > exit
```

> Goal: *one* climate plot, *one* 3D, *one* building sim, *one* PV KPI, *one* sweep, *one* batch — in under two minutes.

Questions?

`python -m batem.baterm` → `help baterm` → start exploring

(`help context` · `help building` · `help pv` · `help enercost` · `manual`)